Approximation Algorithm for Minimum Cost Flow allocation with Varied Survivability

Samer Lahoud, Géraldine Texier, Laurent Toutain

Approximation Algorithm for Minimum Cost Flow Allocation with Varied Survivability

Samer Lahoud, Géraldine Texier and Laurent Toutain
GET/ENST Bretagne
Département Réseaux et Services Multimédia
2 rue de la Châtaigneraie - CS 17607 - 35576 Cesson Sévigné Cedex - France
samer.lahoud@enst-bretagne.fr, geraldine.texier@enst-bretagne.fr, laurent.toutain@irisa.fr

***Abstract*— In this work, we study the minimum cost flow allocation problem with varied survivability. Given a set of demands and a capacitated network, the problem consists of allocating each demand to a set of primary paths that carry the flows realizing the demand volume in the normal operation mode. To ensure survivability, bandwidth is allocated over a disjoint set of backup links protecting the primary path. With the varied survivability concept, only a varied portion of the primary flow is guaranteed to be recovered in failure situations. The recovery ratio is predefined for a given demand and denotes the guaranteed quality of protection. We associate a unitary cost for using the installed bandwidth in core links. Thus, the problem provides a minimum cost solution for allocating the demands and ensuring their survivability. The main contribution of this paper is a new approximation algorithm using a primal-dual approach. This approximation algorithm computes a solution that is within a guaranteed factor of the optimal one and runs in time polynomial in the problem size.**

I. Introduction

Providing minimum cost solutions is a crucial objective for many optimization tasks that are involved in the network design process. Particularly, network engineers solicit flow allocation solutions that induce minimum costs in terms of bandwidth consumption. This cost is strictly related to the capital expenditures spent at the capacity provisioning stage. In this work, we address the problem of finding a minimum cost flow allocation solution. Flow allocation refers to the routing and bandwidth allocation problem for a given set of demands over a capacitated network. Our approach is an off-line optimization that assumes the knowledge of a traffic demand matrix. This matrix defines the bandwidth requirements between network nodes and is generally based on statistical information. We provide support for the varied survivability concept where the backup capacity enables to recover a predefined portion of the primary traffic affected by failures. The recovery ratio is predefined for a given demand and denotes the guaranteed quality of protection.

In this paper, we provide a mathematical formulation for the problem as a Linear Program (LP). We study the complexity of the separation oracle and derive a new approximation algorithm. We prove that the solution cost computed by our approximation algorithm is within a guaranteed factor of the optimal cost and that the algorithm runs in time polynomial in the problem size. This paper is organized as the following. In section II, we state the key elements of this work. We introduce the problem formulation in section III. Then we study the complexity of the problem and the relevant approaches in section IV, emphasizing the complexity of the oracle subproblem. Then, we give the approximation algorithm in section V. Finally, we present the numerical results in section VI, where we evaluate the performance of our algorithm on a typical network topology. We conclude in section VII with a brief summary of the achievements and the open questions encountered in this paper.

II. Context and Motivation

The flow allocation problem with varied survivability includes many key elements that make the corresponding optimization more challenging. In this section, we review the different concepts involved in flow allocation and motivate our approach.

The protection we provide in our work follows the varied survivability concept used in [1]. The basic idea is to offer different levels of protection for different customers while still providing best-effort services. For instance, a service classification can provide (i) guaranteed service quality under normal network operating conditions, i.e. no failure situations, (ii) full guarantee under normal situations plus a reduced level of service under failure situations and (iii) full guaranteed service both under normal or failure situations. Technically, we implement the varied survivability in our problem using a varied protection ratio for each demand: the backup bandwidth protecting the active path of the cycle is computed as a portion of the active bandwidth, and this portion varies form 0% to 100%.

We use a general technique based on duality in LP in order to develop an approximation algorithm for the MCost-MCF problem. In essence, using duality, an optimization problem $\mathcal{P}$ given in a particular form, called *primal problem*, can be transformed to a related problem called its *dual problem*, so that the optimal solutions to both problems are closely related. Particularly, considering that $\mathcal{P}$ is a maximization problem, the objective value of any feasible solution to the primal problem is at most the objective value of any feasible solution of its dual. Hence, when these solutions are equal then they are optimal. Therefore, by comparing the objective values of the primal and dual formulations, we can measure the closeness of a solution (for the primal problem) to the optimal one. The

$\mathcal{V}$	set of vertices
$\mathcal{E}$	set of undirected links
$\mathcal{D}$	set of demands
$\mathcal{K}_d$	set of cycles realizing demand d
C_e	capacity of link e
h_d	volume of demand d
ϑ_d	partial protection ratio for demand d
ξ_e^p	unitary cost for allocating primary flow over link e
ξ_e^b	unitary cost for allocating backup flow over link e
α_{edk}^p	binary constant that equals 1 if link e belongs to the primary path of cycle k; 0, otherwise
α_{edk}^b	binary constant that equals 1 if link e belongs to the backup path of cycle k; 0, otherwise
f_{dk}	flow allocated to cycle k realizing demand d

TABLE I

BASIC PROBLEM NOTATION

approach we present in this paper is particularly innovative in the sense that it provides fast and efficient solutions to the minimum cost problem:

- for any predefined $0 < \rho < 1$, our algorithm computes a solution that is within a $\rho/2$ factor of being optimal
- and our algorithm runs in time polynomial in the network size and the number of demands.

Our contribution is even more relevant since solving directly the initial problem is time consuming in practice and the best practice methods does not perform better (more details are provided in section VI)

III. MATHEMATICAL FORMULATION

In the following, we consider a capacitated network represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. All the notation is described in Table I. Using this notation, we provide a mathematical formulation for the flow allocation problem, referred to as MCost. This formulation is a Linear Program (LP) described in LP 1. Note that in our approach, we use the cycle formulation, where one cycle refers to a pair of disjoint paths. Using disjoint paths ensures the resistance to single link failures. As we allow splittable flows, the demand volume for one demand can be allocated to more than one cycle. Note also that for each unit of primary flow allocated on the primary path of one cycle realizing demand d, a ϑ_d fraction is allocated on the corresponding backup set of links. Particularly, the objective in (1) consists of minimizing the total cost of allocating the demands. It takes into account the two unitary costs of allocating bandwidth for primary and backup flows. Moreover, constraint (2) ensures that the total demand volume for each demand d is allocated on cycles $k \in \mathcal{K}_d$, while constraint (3) ensures that the total flow on link e does not exceed its capacity.

Now let us proceed to a basic reformulation of MCost in order to obtain an elementary problem similar to the problems reported in [2]. The MCost problem formulated in LP 1 consists of minimizing the cost of allocating all the demands subject to capacity constraints. Thus, MCost can be reformulated such that to maximize the portion of allocated demands subject to capacity constraints and a maximal cost bound, and then finding the minimum cost bound that enables to allocate all the demands (details are provided in section V-C). We note that the MCost formulation in LP 1 is a generalization of the minimum cost multicommodity flow problem. Now, we provide in LP 2 an equivalent maximum concurrent flow formulation with a cost bound. This will be referred to as MCost-MCF. In LP 2, constraint (6) ensures that a minimum portion of all demands is allocated to the network: this minimum value is denoted by λ and is subject to maximization in the objective (5). Moreover, constraint (8) denotes the maximum cost, denoted by S, that bounds the solution of allocating a portion of the demands. Finally, constraint (7) denotes the capacity limit on each link.

LP 1 MCost problem

$$\text{Minmize} \quad \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}_d} \sum_{e\in\mathcal{E}} (\alpha_{edk}^p \xi_e^p + \alpha_{edk}^b \xi_e^b \vartheta_d) f_{dk} \tag{1}$$

Subject to:

$$\sum_{k\in\mathcal{K}_d} f_{dk} \geq h_d, \quad \forall d \in \mathcal{D} \tag{2}$$

$$\sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}_d} (\alpha_{edk}^p + \alpha_{edk}^b \vartheta_d) f_{dk} \leq C_e, \quad \forall e \in \mathcal{E} \tag{3}$$

$$f_{dk} \geq 0, \quad \forall d \in \mathcal{D}, \forall k \in \mathcal{K}_d \tag{4}$$

LP 2 MCost-MCF problem

$$\text{Maximize} \quad \lambda \tag{5}$$

Subject to:

$$\sum_{k\in\mathcal{K}_d} f_{dk} \geq \lambda h_d, \quad \forall d \in \mathcal{D} \tag{6}$$

$$\sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}_d} (\alpha_{edk}^p + \alpha_{edk}^b \vartheta_d) f_{dk} \leq C_e, \quad \forall e \in \mathcal{E} \tag{7}$$

$$\sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}_d} \sum_{e\in\mathcal{E}} (\alpha_{edk}^p \xi_e^p + \alpha_{edk}^b \xi_e^b \vartheta_d) f_{dk} \leq S \tag{8}$$

$$f_{dk} \geq 0, \quad \forall d \in \mathcal{D}, \forall k \in \mathcal{K}_d \tag{9}$$

In order to design an approximation problem, let us begin by introducing the dual of the MCost-MCF problem in LP 3. In the dual problem, π_d, ω_e and ϕ denote the dual variables associated with constraints (6), (7) and (8) respectively in the primal problem. In order to follow the same approach adapted from [2], we consider that the generalized weight of one cycle is given by the left-hand-side expression of constraint (12). Then, we need to find an oracle that computes the minimum weight cycle. Unfortunately, this appears as a difficult task, and the related problem deserves a loser look. In the following, we start by reviewing the relevant approaches to the MCost-MCF problem, then we point out the complexity of the oracle in

order derive our approximation.

LP 3 Mcost-MCF: dual problem

$$\text{Minmize} \quad \phi S + \sum_{e \in \mathcal{E}} C_e \omega_e \tag{10}$$

Subject to:

$$\sum_{d \in \mathcal{D}} h_d \pi_d \geq 1 \tag{11}$$

$$\sum_{e \in \mathcal{E}} \alpha^p_{edk}(\omega_e + \xi^p_e \phi) + \alpha^b_{edk} \vartheta_d (\omega_e + \xi^b_e \phi) \geq \pi_d, \quad \forall d \in \mathcal{D}, \forall k \in \mathcal{K}_d \tag{12}$$

$$\omega_e \geq 0, \quad \forall e \in \mathcal{E} \tag{13}$$

$$\phi \geq 0 \tag{14}$$

IV. Complexity Study and Oracle Approximation

A large set of optimization tools can be used in order to provide efficient solutions to the MCost problem. Notably, a similar problem is studied in [1] in the context of network traffic engineering problem for tunnel provisioning. The authors provide a cycle formulation that enables to easily model the varied protection level. The allocation is constrained by bandwidth and tunnel constraints. In order to provide integral solutions to the problem, the authors use two heuristics. The first one consists of solving continuous relaxations of the problem ans then successively rounding the variables to the nearest integer. The second one uses a stochastic meta-heuristic approach by extending the simulated allocation approach introduced in [3]. Another approach consists naturally in adapting the column generation approach as in [4], where no polynomial oracle was found. However, this approach requires maintaining a large set of candidate cycles and thus suffers from computational complexity (as we see in the numerical results in section VI). Therefore, the complexity of the problem is somehow related to the complexity of the oracle. Then, let us investigate the complexity of the oracle in order to get deeper insights on the related problem.

A. Oracle Discussion

1) General Formulation: In order to find a most violated constraint, an efficient subroutine, called oracle, computes the cycle that minimizes the left-hand-side expression of constraint (12) for a given demand d. This is equivalent to finding the minimum weight cycle realizing demand d under the weight function $\omega^p_e = (\omega_e + \xi^p_e \phi)$ for the links in the primary path and $\omega^b_e = \vartheta_d(\omega_e + \xi^b_e \phi)$ for the links in the backup path. Note that link costs ξ^p_e and ξ^b_e are related to the capital expenditures in the capacity provisioning phase. Moreover, our problem supports cost differentiation: different unitary costs can be chosen for bandwidth units whether it is used for a primary or backup flow. Typically, we choose the cost for backup bandwidth to be lower as this bandwidth can be used to carry Best-Effort preemptible traffic. Thus, at his level, we make the assumption that $\xi^p_e \geq \xi^b_e, \forall e \in \mathcal{E}$. Since $0 \leq \vartheta_d \leq 1$, we get $\omega^p_e \leq \omega^b_e, \forall e \in \mathcal{E}$. As we will see in the following study, this assumption plays an important role in the oracle design.

2) Related Work: A similar problem that consists of finding a pair of disjoint paths with minimum weight has been largely investigated recently. In [5], the authors present a general review of the relevant work. They summarize in the table in section 2 the computational complexity of various problems related to finding a pair of disjoint paths. Particularly, the problem of computing a minimum weight disjoint path pair with different weights for the primary and backup paths, called *dual* weights, is studied in [6] and proved to be $\mathcal{N}P$-complete. A special case is to have *ordered* dual weights, e.g. $\omega^p_e \geq \omega^b_e, \forall e \in \mathcal{E}$. This problem of ordered dual weights is studied in [7] and also proved to be $\mathcal{N}P$-complete. Even when weights are uniformly ordered, i.e. there exists a constant $0 < \kappa < 1$ such that for all links we have $\omega^b_e = \kappa \omega^p_e, \forall e \in \mathcal{E}$, the problem is still $\mathcal{N}P$-complete. For directed graphs, the proof is provided in [8], while the proof for undirected graphs is provided in [5].

3) Efficient Algorithms: Given the $\mathcal{N}P$-completeness of the above presented problems, the idea is to find a procedure that efficiently computes a good solution. However, when reviewing the work oriented in this direction, we realize that no existing approach suits our needs for the oracle. In [6], an algorithm with a worst-case guarantee is provided for the arbitrary dual weights problem. The idea is to average the weights on the links and then compute a disjoint pair of paths using the averages weights. Particularly, the algorithm computes a solution that is within a guaranteed factor of the optimal one. The factor depends on the maximal ratio between ω^p_e and ω^b_e. Take for instance, the simple case where $0 \leq \vartheta_d \leq 1, \forall d \in \mathcal{D}$ and $\xi^p_e = \xi^b_e, \forall e \in \mathcal{E}$, then this ratio depends on $1/\vartheta_d$ and the approximation factor is large for small ϑ_d values. Thus, this algorithm is not suitable as an oracle. The authors of [9] consider the same problem and provide an approximation algorithm based on a sophisticated primal-dual approach. Particularly, they formulate the problem as an integer linear program. Then, they relax the integrality and generate a lower bound of the optimal solution. Hence, the algorithm consists of tightening the lower and upper bounds for the problem and evaluating the guaranteed approximation ratio. The authors claim that their algorithm runs fast and the quality of the obtained approximation is less 5%. However, no guarantee for the computation time, nor the number of iterations, is given. Therefore, this approach is not fully suitable for our guaranteed approximation with a desired polynomial running time.

4) Oracle Approximation: After this discussion, we conclude that finding a good oracle that computes a minimum weight pair of paths with dual ordered weights is not an easy task. Therefore, we introduce a new approximation to the oracle with dual ordered weights. This approximation achieves a guaranteed optimality factor of 2 and runs in time

polynomial in the problem size. However, we note that this approximation will induce a structural modification on the basic problem formulation. Let us start by describing the approximation. Our approach is based on the famous algorithm in [10]. Algorithm 1 begins by computing a shortest path form source to destination using weights $\omega_e^p, \forall e \in \mathcal{E}$. This will be the primary path of the output. Then, we construct a residual graph: Each link e on the computed shortest path is replaced by a directed link oriented toward the source and a zero weight. Each other link e is replaced by two directed links in opposite directions with the same weight ω_e^b. On the residual directed graph, we compute a shortest path denoted by q' form source to destination. Then, going back to the original graph, we remove the interlacing links between q' and the primary path and obtain the backup set of links q. Finally, the weight of the set of links in (p, q) is at most two times the weight of the minimum weight cycle realizing demand d. The formal approximation result is provided in Theorem 1.

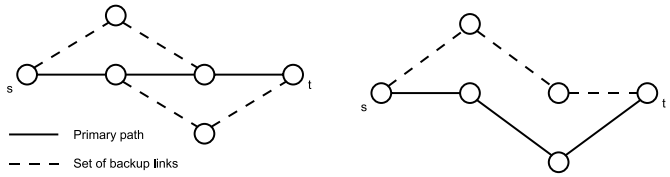
Theorem 1: Algorithm 1 computes a 2-approximation for the minimum weight cycle problem with ordered dual weights.

Proof: Let k^* denote the minimum cycle realizing demand d and let ζ_{k^*} be its weight, given by $\zeta_{k^*} = \sum_{e\in\mathcal{E}} \alpha^p_{edk^*}\omega^p_e + \alpha^b_{edk^*}\omega^b_e = \omega_{p^*} + \omega_{b^*}$, where ω_{p^*} (respectively ω_{b^*}) denotes the weight of the primary path (respectively, the backup path) of the minimum weight cycle. According to Algorithm 1, p is the minimum weight path realizing demand d, with respect to weights ω_e^p. Therefore, the weight of path p, denoted by ω_p, verifies $\omega_p \leq \omega_{p^*} \leq \zeta_{k^*}$. Moreover, considering the algorithm in [10]: the aim of the second step is to compute a minimum cost path on the residual graph (obtained by removing the forward edges on path p). This is a minimum cost flow problem and any cycle is a feasible solution to this problem. Particularly, k^* is a feasible solution, which gives us $\omega_{q'} \leq \sum_{e\in\mathcal{E}}(\alpha^p_{edk^*} + \alpha^b_{edk^*})\omega^b_e \leq \sum_{e\in\mathcal{E}} \alpha^p_{edk^*}\omega^p_e + \alpha^b_{edk^*}\omega^b_e$ (using the assumption that $\omega_e^b \leq \omega_e^p,\ \forall e \in \mathcal{E}$). Hence $\omega_q \leq \omega_{q'} \leq \zeta_{k^*}$, which finally gives us that the weight ζ_k of $k = (p, q)$ verifies $\zeta_k = \omega_p + \omega_q \leq 2\zeta_{k^*}$. ■

However, we note that the output of Algorithm 1 is not a cycle! In fact, when pruning the interlacing links, we get a set of backup links and not a disjoint backup path. Still, this set of backup links ensures the survivability in single failure situations. Thus, we consider that we use obtain a generalized form of protection and we call the pair of primary path and the set of disjoint backup links a *survivable structure* or simply a structure. To some extent, the structure and the generalized protection concept were used in a similar approach to our work, introduced in [11]. The approach formulates a backtracking concept for on-line routing of bandwidth guaranteed tunnels i.e. for a single commodity problem. Backtracking enables to bound the protected entity scope. Particularly, in the no backtracking case, the backup path must originate at the node at which the failed link originates. This case represents true local restoration. In the bounded backtracking case, the backup path can originate at a node on the primary path up to a predefined number of hops away from the node that owns the link. Moreover, in the infinite backtracking case, unlimited backtracking is allowed and therefore, in the worst case may result in backup paths that originate at the source node. The end-to-end restoration can be considered a special case of this where the maximum backtracking allowed is equal to the length of the primary path. The last case is very similar to the generalized approach we introduce in our work. The authors in [11] show that joint optimization of primary and backup paths is $\mathcal{NP}$-hard in all cases and they consider algorithms that compute primary and backup paths in two separate steps. In the following we formalize the definition of the output structure of the oracle approximation.

input : graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, demand $d = (s_d, t_d)$, $\{\omega_e^p,\ \forall e \in \mathcal{E}\}$ and $\{\omega_e^b,\ \forall e \in \mathcal{E}\}$
output : approximate minimum weight structure $k \in \mathcal{K}_d$
assume : $\omega_e^p \geq \omega_e^b,\ \forall e \in \mathcal{E}$
foreach $e \in \mathcal{E}$ **do** assign a weight ω_e^p;
$p \leftarrow$ shortest path realizing demand d;
foreach $e \in p$ **do**
  replace e with a directed link oriented toward the source and let the weight of this link be zero;
foreach $e \in \mathcal{E} - p$ **do**
  replace e with two directed links in opposite directions and let the weight of these links be ω_e^b;
$q' \leftarrow$ shortest path realizing demand d in the modified directed graph;
$q \leftarrow$ {set of links in the original graph that are in q' and not in p};
$k \leftarrow (p, q)$;

Algorithm 1: Oracle Approximation



(a) Primary path and disjoint set of backup links

(b) Two link-disjoint paths

Fig. 1. Examples of survivable structures

B. Structures and Cycles

We have noted above that a structure obtained by Algorithm 1 consists of a primary path going from source to destination and a set of backup links that are disjoint from the primary path and ensure the survivability. More formally, a structure k realizing demand $d = (s_d, t_d)$ is a set of links that consists of a primary path p from s_d to t_d and a set of backup links b the are disjoint from p. For each single failure situation the set $(p \cup b)$ contains a path from s_d to t_d. Thus, in a structure, backup links together with the primary links that are not failing are required to provide connectivity from source to destination in failure situations.

Mathematically, we can define a structure using basic properties of graph theory [12]. Recall that an $(s-t)$ cut, denoted by $\mathcal{C}_{(s-t)}$, in graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a partition of the set of vertices $\mathcal{V}$ into disjoint sets $\mathcal{S}$ and $\mathcal{T} = \mathcal{V} - \mathcal{S}$, such that $s \in \mathcal{S}$ and $t \in \mathcal{T}$. The links of the cut are those that have precisely one endpoint in both $\mathcal{V}$ and $\mathcal{T}$. Let us denote the set of links of the cut by $\mathcal{E}_{\mathcal{C}_{(s-t)}}$. Therefore, a structure k realizing demand $d = (s_d, t_d)$ verifies that there are more than 2 links belonging to $k \cap \mathcal{E}_{\mathcal{C}_{(s_d-t_d)}}$, for any (s_d, t_d)-cut $\mathcal{C}_{(s_d-t_d)}$. In Figure 1, we provide two examples that enable to illustrate the new concept of structures. Figure 1(a) is a general case of a structure obtained by the oracle approximation. However, in some cases and depending on the output of the approximation algorithm, a structure consists of two disjoint paths, as in Figure 1(b). Thus, we consider that the cycle formulation is a special case of the structure formulation.

V. Approximation Algorithm

Now that we have an approximate oracle, we continue the design of the global approximation to the MCost-MCF problem by adapting the approach introduced in [13].

A. Algorithm Presentation

Algorithm 2 computes an approximation to the MCost-MCF problem. This algorithm works as follows: we start with initial link weights $\omega_e = \frac{\delta}{C_e}, \forall e \in \mathcal{E}$ and $\phi = \frac{\delta}{S}$. The quantity δ depends on the approximation guarantee (particularly on a parameter ϵ introduced later) and will be stated later in the algorithm analysis section. The algorithm proceeds in phases. Each phase is composed of D iterations. In iteration d, we allocate h_d units of flow for demand d. This flow is allocated in a sequence of steps. In each step, a cycle k^* (realizing demand d) that minimizes the left-hand-side expression of constraint (12) should be computed. However, as we have shown in the oracle study we can compute an approximate structure denoted by k. Now, we will compute the flow we allocate to this structure. Considering the varied survivability concept, k consists of a primary path p and a backup set of links b with bottleneck capacities denoted by c_p and c_b respectively. The bottleneck capacity of structure k is given by the minimum of (i) c_p, and (ii) c_b/ϑ_d. This is due to the fact that we allocate a ϑ_d fraction of bandwidth on the backup links for each 1 unit of flow on the primary path. Therefore, the maximum amount of flow allocated to structure k in this step is denoted by c and is the minimum of:

1) the bottleneck capacity of structure k given by $\min\{c_p, c_b/\vartheta_d\}$,
2) the remaining amount of flow that need to be allocated for demand d to make a total of h_d.

However, we should ensure that the cost of allocating c units of flow to structure k does not exceed S, otherwise the flow should be scaled appropriately. Particularly, let us denote by s_k the cost of allocating 1 unit of flow to structure k. Recall that this is equivalent to allocating 1 unit to the primary path and a ϑ_d fraction of unit to the backup links. Thus, $s_k = \sum_{e\in\mathcal{E}}(\alpha^p_{edk}\xi^p_e + \alpha^b_{edk}\xi^b_e\vartheta_d)$. Similarly, the cost of allocating c units of flow to structure k is given by cs_k. Therefore, we ensure that:

1) if $cs_k > S$, we scale c according to $c = c\frac{S}{s_k}$,
2) if else, the value of c remains unchanged.

Finally, we get a value of c that denotes the amount of flow to be allocated to the structure k computed in this step. The primal variables are updated correspondingly and the dual weights are updated as follows (using the parameter ϵ that will determine the approximation ratio):

1) for each $e \in p$, $\omega_e = \omega_e(1 + \epsilon\frac{c}{C_e})$,
2) for each $e \in b$, $\omega_e = \omega_e(1 + \epsilon\frac{\vartheta_d c}{C_e})$,
3) and $\phi = \phi(1 + \epsilon\frac{cs_k}{S})$.

The algorithm terminates when the dual objective function value becomes bigger than one. At termination, dual feasibility constraints will be satisfied. However, link capacity constraints in the primal solution will be violated, since we use the original (and not the residual) link capacities at each stage. The full details are provided in the following section, where we study the global performance of the algorithm and the corresponding approximation ratio.

input : Capacitated graph $\mathcal{G}$, set of demands $\mathcal{D}$, cost bound S and ϵ
output : $\tilde{\lambda} = \frac{t-1}{\log_{1+\epsilon}(\frac{1}{\delta})}$, $\tilde{S} = \frac{s}{\log_{1+\epsilon}(\frac{1}{\delta})}$
initialize: $t = 0$; $s = 0$; $\omega_e = \frac{\delta}{C_e}, \forall e \in \mathcal{E}$; $\phi = \frac{\delta}{S}$;
while $\Omega < 1$ **do**
  for $d = 1, \ldots, D$ **do**
    $r \leftarrow h_d$;
    while $\Omega < 1$ *and* $r > 0$ **do**
      $k = (p, b) \leftarrow$ structure realizing demand d given by Algorithm 1;
      $c \leftarrow \min\{r, c_p, c_b/\vartheta_d\}$;
      $s_k \leftarrow \sum_{e\in\mathcal{E}}(\alpha^p_{edk}\xi^p_e + \alpha^b_{edk}\xi^b_e\vartheta_d$;
      if $cs_k > S$ **then** $c \leftarrow \frac{c}{s_k}$;
    $s \leftarrow s + cs_k$;
    $\forall e \in p : \omega_e \leftarrow \omega_e(1 + \epsilon\frac{c}{C_e})$;
    $\forall e \in b : \omega_e \leftarrow \omega_e(1 + \epsilon\frac{\vartheta_d c}{C_e})$;
    $\phi \leftarrow \phi(1 + \epsilon\frac{cs_k}{S})$;
    $r \leftarrow r - c$;
  $t \leftarrow t + 1$;

Algorithm 2: MCost-MCF approximation algorithm

B. Algorithm Analysis

Let us denote by $\omega_e(i, d, j-1)$ the weight of link e and $\phi(i, d, j-1)$ the value of ϕ at the beginning of step j of iteration d in phase i. Let $k^*(i, d, j-1)$ denote the minimum weight cycle in $\mathcal{K}_d$ in step j of iteration d of phase i, using weights $\omega_e(i, d, j-1)$, and let $\zeta_{k^*(i,d,j)}(i, d, j-1)$ be its weight. We know that the weight of the computed structure $\zeta_{k(i,d,j)}(i, d, j-1)$ does not exceed $2\zeta_{k^*(i,d,j)}(i, d, j-1)$. Let also $\Omega(i, d, j)$ be the objective of the dual problem LP 3 at the end of the iteration. Assuming that there are $J = J(i, d)$

steps executes at iteration d of phase i, let us consider that $\Omega(i) = \Omega(i, D, J)$ and $\Gamma(i) = \sum_{d=1}^{D} h_d \zeta_{k^*(i,d)}(i, D)$, where $\zeta_{k^*(i,d)}(i, D)$ denotes the weight of the minimum weight cycles (computed originally with respect to $\omega_e(i, d-1, J)$ and $\phi(i, d-1, J)$) at the end of phase i. Then, the dual problem is an assignment of weights ω_e to links $e \in \mathcal{E}$ and a scalar ϕ, which we view as a weight associated with a pseudo-link of capacity S, such that $\frac{\Omega(i)}{\Gamma(i)}$ is minimized. Moreover, the optimal dual objective is given by $\psi := \min_i \frac{\Omega(i)}{\Gamma(i)}$. In the algorithm analysis, we assume that $\psi \geq 1$ but we shall remove this assumption later after.

We start by computing the ratio between the optimal dual objective ψ and the primal objective obtained at the end of the algorithm. Knowing that we allocate h_d units of flow for each demand d in each phase, then the primal objective is given by $t-1$, where t denotes the total number of phases. Lemma 1 gives the corresponding intermediate result. Note that this primal solution does not necessarily correspond to a feasible solution, and thus need to be scaled properly in order to become feasible. Therefore, Lemma 2 and Lemma 3 give the proper scaling that yields a feasible solution to the primal problem and the corresponding cost. For clarity, the full mathematical proof is left for the Appendix. Hereafter, we provide the main results of the algorithm analysis.

Lemma 1: At the end of Algorithm 2, the ratio between the optimal dual objective and the number of phases verifies $\frac{\psi}{t-1} \leq \frac{2\epsilon}{(1-2\epsilon)\ln\frac{1-2\epsilon}{\delta(E+1)}}$.

Lemma 2: At the end of Algorithm 2, $\tilde{\lambda} = \frac{t-1}{\log_{1+\epsilon}(\frac{1}{\delta})}$ yields a feasible primal solution.

Lemma 3: At the end of Algorithm 2, $\tilde{S} = \frac{s}{\log_{1+\epsilon}(\frac{1}{\delta})}$ is the cost of the feasible primal solution.

Theorem 2: Assuming that $0 < \rho < 1$, $\epsilon = \frac{1-\rho^{\frac{1}{3}}}{2}$ and $\delta = (\frac{E+1}{1-2\epsilon})^{-1/\epsilon}$, Algorithm 2 computes a $\rho/2$-approximation to the MCost-MCF problem in time $O\Big((2D\log D + E + 1)\left\lceil \frac{1}{\epsilon} log_{1+\epsilon}\frac{E+1}{1-2\epsilon}\right\rceil T_{mws}\Big)$, where T_{mws} is the time required to compute a minimum weight structure.

The running time for computing a minimum weight structure as in Algorithm 1 is dominated by two minimum weight path computation that can be performed in $O(E + V\log V)$ [14].

C. Finding the Minimum Cost Solution

With a $\rho/2$-approximation to the MCost-MCF problem, we can obtain a $\rho/2$-approximation to the MCost problem introduced in LP 1. For this purpose, we can use an efficient search method such as binary search to find the smallest value of S for which MCost-MCF has a solution of $\tilde{\lambda} \geq 1$. Furthermore, the binary search can be improved by using an interpolation search as in [15]. This improvement consists of estimating the next value using a linear interpolation as in the following:

- if MCost-MCF with a bound S returns a value $\tilde{\lambda} \geq 1$, then the final cost bound is at most $S/\tilde{\lambda}$;
- if MCost-MCF with a bound S returns a value $\tilde{\lambda} \leq 1$, then the final cost bound is at least $S/\tilde{\lambda}$.

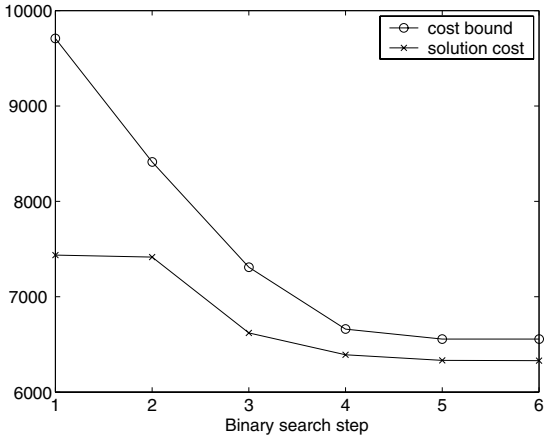


Fig. 2. Cost evolution

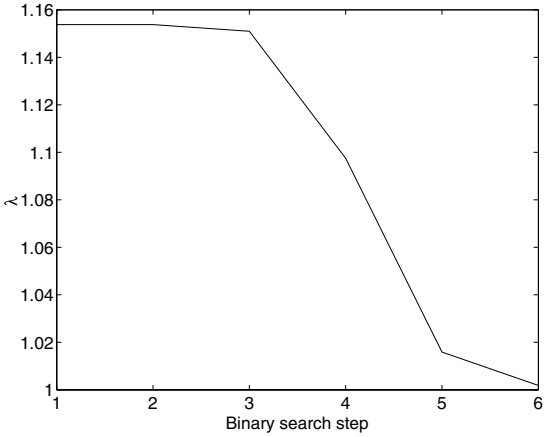


Fig. 3. Objective of MCost-MCF

This interpolated search enables to quickly converge to the smallest value of S for which MCost-MCF has a solution of $\tilde{\lambda} \geq 1$. This value is a $\rho/2$-approximation to the MCost problem. Note finally that a simple upper bound for the solution cost, given for instance by $\sum_{e\in\mathcal{E}} \xi_e^p C_e$, can be used to start the binary search.

VI. Numerical Results

We present a typical numerical result corresponding to the KL topology [9]. KL is a 15-node 28-link topology. We associate a capacity of 200 bandwidth units with each link and consider that demands are defined between each node pair of the network, each having have a volume of 10 bandwidth units. We associate a unitary cost of 2 for the links in the primary path and 1 for the links in the backup path. For this typical simulation, the binary search ends in 6 steps. Note that we start with the simple upper cost bound as indicated in section V-C. In Figure 2, we evaluate the cost of the solution for the MCost-MCF problem throughout the binary search steps. Let us analyze simultaneously Figure 3 that gives the value of λ (the objective of the MCost-MCF problem) during the binary search process. At the first two steps, the cost bound is very

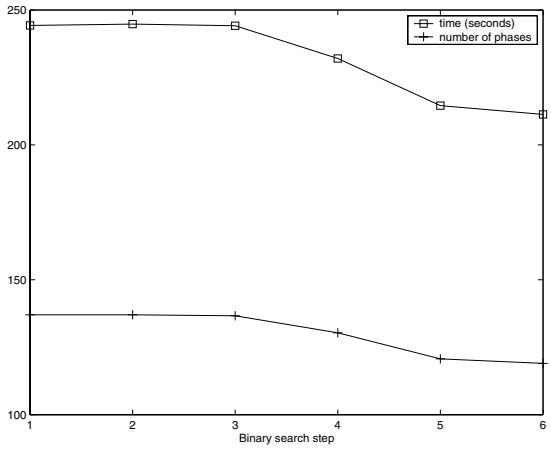


Fig. 4. Time performance

loose, and the value of λ is around 1.15. Then, using the linear interpolation we start strengthening the bound in order to make λ converge toward 1. Thus, in the third step, the cost of the solution goes down from 7500 to 6500, while λ is still around 1.15. This is due to the fact that the solution of the problem is not unique and two allocations of the same bandwidth volume can lead to different costs. However continuing the interpolated binary search, we obtain two steps further (step 5) a solution with λ around 1.01 and a cost bound of 6500. We note that at this level, smaller reductions in the cost bound has bigger impact on the value of λ. Finally at the sixth step, the desired accuracy is reached for a cost of 6300 and λ around 1.0001.

Finally, in Figure 4, we assess the time performance of the algorithm. We provide for each binary step the number of phases, and the total time spent in the algorithm. We can see that the total number of phases in each run of the algorithm decreases with the steps. This is due to the fact that λ is decreasing and less flow is allocated, then a smaller number of phases is needed. Similarly, the running time is decreasing with the binary steps. An important remark is that the total running time (the sum for the six steps) is very competitive with the running time for the column generation approach. A complete study and further comparison is provided in [16].

VII. Conclusion

The advent of Next Generation Internet (NGI) networks has launched many services related to survivability. Therefore, the need for efficient algorithmic tools that enable network engineers to implement their provisioning strategies is becoming urgent. In this work we have designed an efficient algorithm for the flow allocation problem. Our problem takes into account varied survivability levels in terms of guaranteed protection and enables to minimize the network costs. An extension of this work encompasses multiple failure survivability is reported in [16]. Moreover, detailed comparison with the best practice methods, especially with column generation, is provided in [16]. Future work will include rounding the fractional solution to get integral flows and comparing the results with stochastic meta-heuristics.

Appendix I
Analysis of Algorithm 2

In the following, we highlight the basic steps in the algorithm analysis. In [16] a complete version is provided.

A. Proof of Lemma 1

As we have noted, Algorithm 2 proceeds in phases: each phase is composed of D iterations. In iteration i, we allocate h_d units of flow of demand d and the flow is allocated in a sequence of steps. Using the notation introduced in section V-B, we have:

$$\begin{aligned}\Omega(i,d,j) &= \phi(i,d,j)S + \sum_{e\in\mathcal{E}} C_e\omega_e(i,d,j)\\ &= \phi(i,d,j-1)(S+\epsilon c s_k) + \sum_{e\in\mathcal{E}} C_e\omega_e(i,d,j-1)\\ &\quad +\epsilon c\sum_{e\in\mathcal{E}} \alpha^p_{edk}\omega_e(i,d,j-1) + \alpha^b_{edk}\vartheta_d\omega_e(i,d,j-1).\end{aligned}$$

Using the fact that $s_k = \sum_{e\in\mathcal{E}}(\alpha^p_{edk}\xi^p_e + \alpha^b_{edk}\xi^b_e\vartheta_d)$, we get:

$$\begin{aligned}\Omega(i,d,j) &= \Omega(i,d,j-1) + \epsilon c\zeta_{k(i,d,j)}(i,d,j-1)\\ &\le \Omega(i,d,j-1) + 2\epsilon c\zeta_{k^*(i,d,j)}(i,d,j-1) \quad (15)\end{aligned}$$

Assume that there are $J = J(i,d)$ steps executed in iteration d in phase i. Adding the inequalities in (15) for demand d of phase i and noting that $\zeta_{k^*(i,d,j)}(i,d,j-1) \le \zeta_{k^*(i,d,j)}(i,d,J)$, we get

$$\Omega(i,d,J) \le \Omega(i,d,0) + 2\epsilon h_d\zeta_{k^*(i,d,j)}(i,d,J).$$

Adding the above inequalities over all demands in phase i, and using the fact that weights are monotonically increasing over iterations, we get

$$\Omega(i) \le \Omega(i-1) + 2\epsilon\Gamma(i).$$

where $\Omega(i,D) = \Omega(i)$. Since $\frac{\Omega(i)}{\Gamma(i)} \ge \psi$, we have

$$\Omega(i) \le \frac{\Omega(i-1)}{1-\frac{2\epsilon}{\psi}}$$

and with $\Omega(0) = \delta(E+1)$, we have for $i \ge 1$

$$\begin{aligned}\Omega(i) &\le \frac{\delta(E+1)}{(1-\frac{2\epsilon}{\psi})^i}\\ &= \frac{\delta(E+1)}{1-\frac{2\epsilon}{\psi}}(1+\frac{2\epsilon}{\psi-2\epsilon})^{i-1}\\ &\le \frac{\delta(E+1)}{1-\frac{2\epsilon}{\psi}}e^{\frac{2\epsilon(i-1)}{\psi-2\epsilon}}\\ &\le \frac{\delta(E+1)}{1-2\epsilon}e^{\frac{2\epsilon(i-1)}{\psi(1-2\epsilon)}}\end{aligned}$$

where we assume that $\psi \ge 1$ in the last inequality. At phase t, we get $\Omega(t) \ge 1$ and the procedure stops. Therefore,

$$1 \le \Omega(t) \le \frac{\delta(E+1)}{1-2\epsilon}e^{\frac{2\epsilon(t-1)}{\psi(1-2\epsilon)}}$$

which implies

$$\frac{\psi}{t-1} \leq \frac{2\epsilon}{(1-2\epsilon)\ln\frac{1-2\epsilon}{\delta(E+1)}}.$$

B. Proof of Lemma 2

Considering edge e and associated weight ω_e. The value of ω_e is updated when flow is allocated on edge e. Particularly, this is done when e belongs to the active or backup set of links of a chosen structure. In either cases, ω_e is updated by a factor of at most $1+\epsilon$. Let the sequence of flow allocation that require an update of ω_e be $\Delta_1, \Delta_2, \ldots, \Delta_l$. Let $\sum_{i=1}^{l} \Delta_i = \kappa C_e$, i.e. the total flow allocated to edge e exceeds its capacity by a factor of κ. Scaling $t-1$ (the allocated portion at the end of the algorithm) by κ lead to a feasible solution (satisfying capacity constraints). Since the algorithm terminates when $\Omega(t) \geq 1$, we have $\Omega(t-1, D, J) < 1$ and $\Omega(t, D) < 1+\epsilon$. Therefore, we have $C_e\omega_e(t-1, D, J) < 1$ and

$$\omega_e(t-1, D, J) = \frac{\delta}{C_e}\prod_{i=1}^{l}(1+\epsilon\frac{\Delta_i}{c_e}).$$

Knowing that $(1+ax) \geq (1+x)^a, \forall x \geq 0$ and any $0 \leq a \leq 1$, we get

$$1 > C_e\omega_e(t-1, D, J) \quad \geq \quad \delta\prod_{i=1}^{l}(1+\epsilon)^{\frac{\Delta_i}{c_e}} = \delta(1+\epsilon)^{\kappa}$$

Therefore $\kappa < log_{1+\epsilon}\frac{1}{\delta}$ and scaling $t-1$ by κ gives the result in the lemma. For scaling the computed value of the total cost s, the proof follows the same sketch as the one provided for λ. Note that $\phi(t-1, D, J) < 1/S$, $\phi(1, 0, 0) = \delta/S$ and that every time we allocate a flow whose total cost is S, we increase ϕ by a factor of $1+\epsilon$.

C. Proof of Theorem 2

Let η be the ratio of the values of the optimum dual and primal solutions. Then, η verifies

$$\eta < \frac{\psi}{t-1}\log_{1+\epsilon}(\frac{1}{\delta}).$$

By substituting the bound value given by Lemma1, we get

$$\eta \leq \frac{2\epsilon}{(1-2\epsilon)\ln\frac{1-2\epsilon}{\delta(E+1)}}\log_{1+\epsilon}(\frac{1}{\delta})$$

For $\delta = (\frac{E+1}{1-2\epsilon})^{-1/\epsilon}$, we obtain

$$\eta \leq \frac{2\epsilon}{(1-\epsilon)(1-2\epsilon)\ln(1+\epsilon)} \leq 2(1-2\epsilon)^{-3}.$$

Therefore, choosing $\epsilon = \frac{1-\rho^{\frac{1}{3}}}{2}$ will lead to a $\rho/2$-approximation algorithm with $0 < \rho < 1$.
By weak duality we have

$$1 \leq \eta \leq \frac{\psi}{t-1}\log_{1+\epsilon}(\frac{1}{\delta})$$

hence the number of phases, t, is strictly less than $1 + \psi\log_{1+\epsilon}(\frac{1}{\delta})$ which implies that $t = \left\lceil \frac{\psi}{\epsilon}\log_{1+\epsilon}\frac{E+1}{1-2\epsilon} \right\rceil$. Following the analysis provided in [13], we can scale the capacities so that $\psi \geq 1$ and the total number of phases is $T\log D$ (refer to [16] for more details). Moreover, in each step, except the last step in an iteration, we increase either the weight of some edge or the value of ϕ by a factor $1+\epsilon$. Hence, the weights of the edges and ϕ can be increased by a factor $1+\epsilon$ at most $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$ times. Therefore the number of steps exceeds the number of iterations by at most $(E+1)\left\lceil \frac{1}{\epsilon}\log_{1+\epsilon}\frac{E+1}{1-2\epsilon} \right\rceil$.

REFERENCES

[1] S. Srivastava, S.R.Thirumalasetty, and D. Medhi, "Network traffic engineering with varied levels of protection in next generation internet," *GERAD 25th Anniversary Issue for Performance Evaluation and Planning Methods for the Next Generation Internet*, 2005.

[2] L. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," *SIAM J. Discret. Math.*, vol. 13, no. 4, pp. 505–520, 2000.

[3] M. Pióro and P. Gajowniczek, "Simulated allocation: a suboptimal solution to the multicommodity flow problem," in *Teletraffic Symposium on Performance Engineering in Telecommunications Networks*, I. Press, Ed., 1994.

[4] B. Krithikaivasan, S. Srivastava, D. Medhi, and M. Pioro, "Backup path restoration design using path generation technique," in *Design of Reliable Communication Networks*, Alberta, Canada,, 2003.

[5] D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He, "On finding disjoint paths in single and dual link cost networks," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 1, 2004, pp. –715.

[6] C.-L. Li, S. McCormick, and D. Simchi-Levi, "Finding disjoint paths with different path costs: Complexity and algorithms," *Networks*, vol. 22, pp. 653–667, 1992.

[7] Y. Liu, "Spare capacity allocation method, analysis and algorithms," Ph.D. dissertation, School of Information Sciences, University of Pittsburgh, 2001.

[8] P. Laborczi *et al.*, "Solving asymmetrically weighted optimal or near-optimal disjoint path-pair for the survivable optical networks," in *Third International Workshop On Design of Reliable Communication Networks, DRCN'01*, October 2001.

[9] M. Kodialam and T. Lakshman, "Dynamic routing of restorable bandwidth-guaranteed tunnels using aggregated network resource usage information," *Networking, IEEE/ACM Transactions on*, vol. 11, no. 3, pp. 399–410, 2003.

[10] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325–336, 1986.

[11] L. Li, M. M. Buddhikot, C. Chekuri, and J. K. Guo, "Routing bandwidth guaranteed paths with local restoration in label switched networks," in *ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 110–121.

[12] R. Diestel, *Graph theory*, 3rd ed. Springer-Verlag, 2005. [Online]. Available: http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/GraphTheoryII.pdf

[13] N. Garg and J. Koenemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems." in *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1998, p. 300.

[14] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, 1987.

[15] L. Fleischer, A. Meyerson, I. Saniee, B. Sheperd, and A. Srinivasan, "A scalable algorithm for the minimum expected cost restorable flow problem," CORC, Tech. Rep. TR-2003-10, 2004.

[16] S. Lahoud, "Survivable routing and flow allocation in next generation internet networks," Ph.D. dissertation, GET/ENST Bretagne, 2006.